|  |  |  |  |
| --- | --- | --- | --- |
| **Qualification details** |  | | |
| **Qualification National Code & Title** | ICT40120 Certificate IV in Information Technology | **State code** | AC07 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Assessment Title**  (as per DAP) | AT2 – Portfolio | | | | |
| **Unit National Code & Title** | ICTPRG430 Apply introductory object-oriented language skills | | | **State code** | OAQ05 |
| **Date Due** | Refer to the DAP | **Date Received** |  | | |

|  |  |
| --- | --- |
| **Student Name & ID** | Brayam Mazuera, 30085203 |
| **Student Declaration** | I declare that the evidence submitted is my own work:  ………………………………………………………….. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Assessor Name** | Ken Beck | | | |
| **Assessment Decision** | * Satisfactory | | * Not Yet Satisfactory | |
| **Is student eligible for reassessment (Re-sit)?** | * Yes | * No | **Reassessment Date** |  |
| **Assessor Signature** |  | | **Date** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Feedback to student** | | | |
|  | | | |
| **Feedback from student** | | | |
|  | | | |
| **Student signature** | Brayam Mazuera | **Date** | 16/08/2024 |

|  |  |
| --- | --- |
| **Candidate Instructions** | |
| The analysis, design, coding, testing and project documentation of a Java application as described on the following page. | |
| **Duration of Assessment** | Week 2 ~ 12 |
| **Location of Assessment** | In class |
| **Reasonable adjustment** | In some circumstances, adjustments to assessments may be made for you. If you require support for literacy and numeracy issues; support for hearing, sight or mobility issues; change to assessment times/venues; use of special or adaptive technology; considerations relating to age, gender and cultural  beliefs; format of assessment materials; or presence of a scribe you need to inform your lecturer. |

|  |
| --- |
| **Resources Required** |
| Windows 10 system environment / Java SDK software package and NetBeans IDE / Oracle Academy Practice files / Blackboard / Internet |

|  |
| --- |
| **Performance Measurement** |
| Portfolio questions that are performed after each session and submitted by the due specified in the DAP |

|  |  |
| --- | --- |
| **Assessor Instructions** | |
| **Type of Assessment** | Per session practice exercises and programming design |
| **Duration of Assessment** | Week 2 ~ 12 |
| **Location of Assessment** | In class or own place |
| **Conditions** | Student works are submitted to Blackboard and feedback is given to student on Blackboard |
| **Marking Checklist** | Refer to the Marking Guide |
| **Due Date** | Refer to the DAP |

|  |
| --- |
|  |
| **Assessment Specification** |
| **NOTE:**   * The job role environment is that you are working as a junior programmer in a software development company. You will develop small-sized applications with Object Oriented Programming knowledge and skills. * There are different types of questions   + If the question requires performing/demonstrating some processes, screenshots are required.     - In class demonstrations will be required for further evidence.   + If programming is required, make sure you include your source files (xxx.java) and provide screenshots of programs running.     - Your programs cannot be assessed without your source files. * Submit your source code with compile/run batch files.   + Provide your source files.   + Provide your compile batch files (e.g. ***compile.bat***).     - This batch file compiles your source and generates .class files.   + Provide your run batch files (e.g. ***run.bat***).     - This batch file runs your application. * Do NOT submit the whole IDE project folders.   + Submit source files and batch files only. * Refer to Java API specification for available classes * Your programs will be tested using Oracle Java JDK version 19 or higher. * If written answering is required, write your answers in text files, e.g. MS-WORD or Notepad. * With all other types of questions, save your answers appropriately. * All programming code must comply with the Java Code Convention. (Your works will be assessed against this)   + Auto Format option is available in NetBeans. * Zip all your files into a single .zip file before submit.   + Save your zip file as the submit name, e.g. Portfolio\_Activity1.zip. * To ensure authenticity, the students may be required to demonstrate in class for assessment. * **Cheating and plagiarism may result in unit re-enrol.**   **Activity 1**   * Task A.   + Demonstrate how to compile and run the *HelloWorld* program below using the JDK javac and java commands in a terminal window, e.g. DOS command window.     - Provide your compile/run batch files. public class HelloWorld {   public static void main(String[] args) { System.out.println("Hello World");  }  }   * Task B.   + Demonstrate how to develop the HelloWorld program using the NetBeans IDE.       **Activity 2**   * Task A.   + Your goal is to create a program similar to *JavaStory* (**JavaStory.class** demo is available). Write a story where certain parts of the resulting story text are modified by the user’s input. Prompt the user for various inputs.   + You accept user inputs using the ***Scanner*** class.   + When you output your story, make sure all your text is visible on the screen. Your story may need to be spread across several lines instead of being printed in one giant line of output.   This helps keep your output clean and your program more user friendly.   * It’s ok for your program to crash if the user inputs inappropriate data. For example, if the user enters a string when an int value is required. Just assume that the user always enters correct types of inputs. We’ll cover exception handling later in this course. * Your program must do the following:   + Accept at least 1 input, to be parsed as a ***String***   + Accept at least 1 input, to be parsed as an ***int***   + Accept at least 1 input, to be parsed as a ***double***   + Use at least 1 input in a question for the user   + Do math with at least 1 ***int*** input   + Do math with at least 1 **double** input * **NOTE**: Apply code and documentation conventions against the Java Code Convention.   + For example, correct indentations, correct position of the start of a block, correct spacing, naming conventions, etc… * Save your main program as ***JavaStory.java*** |

|  |  |  |
| --- | --- | --- |
| * Task B.   + Create a program that receives three numbers and displays the total and the average of the numbers. Use appropriate data types, e.g. int for the total and double for the average. Use correct operators and expressions in order to create clear and concise code in your program.   + The three numbers from the user must be stored in **int** variables.   + The program must calculate the average correctly in a decimal number.   + Save the main program as ***TotalAverage.java***          * Task C.   + Write an application that inputs one number consisting of five digits from the user, separates the number into its individual digits and prints the digits separated from one another by spaces each. For example, if the user types in the number 42379, the application should print.   4 2 3 7 9   * + Do not use conditional if nor loops (e.g. for) for this question nor arrays, etc. Use the basic 5 arithmetic operators (+ - \* / %) only for each digit separation.   + Save the main program as ***NumberWithSpaces.java***       **Activity 3**   * Task A.   + Write an interactive Java program, ***ColorRange.java***, which when given a wavelength in nanometres will return the corresponding colour in the visible spectrum.   **Colour Wavelength (nm)**  Violet 380-450  Blue 450-495  Green 495-570  Yellow 570-590  Orange 590-620  Red 620-750   * + You must implement the following using ***if*** statements.     - Prompt the user to enter the wavelength, the wavelength should be of type ***double***.     - For each range (e.g. 380-450) the number on the left is included in the range, but the number on the right is not included in the range.     - If the input value is not found on the visible spectrum then state that the wavelength is not within the visible spectrum.   + A sample output is as below: | | |
|  | Enter a colour code 630  The colour is Red |  |
| Enter a colour code 25.0 |  |

|  |
| --- |
|  |
|  |
|  |
|  |

|  |  |  |
| --- | --- | --- |
|  | The entered wavelength is not a part of the visible spectrum |  |
| Enter a colour code 750.5  The entered wavelength is not a part of the visible spectrum |  |
| * Task B.   + The normal behaviour for a stop light is to cycle from Red to Green to Yellow to Red (and continues with this pattern). Write a java program ***StopLight.java***, which will determine the next colour of a stop light in this pattern, Red to Green to Yellow to Red based on the current stop light provided by the user.      * + You must implement the following using ***switch*** statements or ***switch*** expressions.     - Have the user enter the value for the currentColor.     - Compute the next colour stop light based on the currentColor.     - Alert the user for any invalid value of colour.   + A sample out is as below:   Enter a colour code (1: red, 2: green, 3: yellow) 1  Next Traffic Light is green  Enter a colour code (1: red, 2: green, 3: yellow) 3  Next Traffic Light is red  Enter a colour code (1: red, 2: green, 3: yellow) 0  Invalid colour  Enter a colour code (1: red, 2: green, 3: yellow) 4  Invalid colour  **Activity 4**   * Task A.   + Develop a java program, ***PinValidator.java***, to validate bank PIN of a customer. Use a ***while*** loop to repeat code until a valid PIN is entered   + You must implement the follow:     - Declare a valid integer PIN.     - Prompt the user to enter the PIN.     - In a ***while*** loop, perform the following steps:       * Compare the user-entered PIN with the already declared PIN.  * + - * If the entered PIN is not correct, prompt the user to enter the PIN again.       * Repeat the loop until the correct PIN is entered.     - Print a message confirming that the correct PIN has been entered and that the user now has access to their account.   + Demonstrate how to use the NetBeans Debugger tool to test/debug this PinValidator program. * Task B.   + Develop a java program, ***MultiTable.java***, that displays the multi-table of a given number using a **for** loop.   + Have the user enter a number, and then use a ***for*** loop to display all the multiples of that number from 1 to 12. | | |

|  |  |  |
| --- | --- | --- |
| * Expected Output: Choose a number: 7   7 X 1 = 7  7 X 2 = 14  7 X 3 = 21  7 X 4 = 28  7 X 5 = 35  7 X 6 = 42  7 X 7 = 49  7 X 8 = 56  7 X 9 = 63  7 X 10 = 70  7 X 11 = 77  7 X 12 = 84  **Activity 5**   * Task A.   + You are asked to develop a java program that performs:     - Converts a temperature of Fahrenheit to Celsius     - Calculates the hypotenuse of a triangle     - Rolls a dice   + Your first step is to design the application     - Create application design specifications document that satisfies user requirements.     - Review and clarify user requirements with your assessor as necessary.     - Save your document file as ***Application Specifications Document.docx***   + Your next step is to implement the application     - Develop application according to your application design and organisational code convention (Java Code Convention)     - Document application with enough comments according to the Java comments convention   + Read the requirements below and complete this task.   + Create a class that defines the following three methods as below: (Refer to the Java API documentation for available Java SDK classes and methods)     - This class will be saved as ***ComputeMethods.java*** | | |
|  | * public double fToC(double degreesF)   + It converts the given temperature in Fahrenheit to Celsius and returns it as double.      * public double hypotenuse(int a, int b)   + It computes the hypotenuse length of a triangle given its side lengths and returns is as double   C:\Users\ken\AppData\Local\Temp\SNAGHTML3df4c2bc.PNG   * public int roll()   + It simulates the rolling of two 6-sided dice and returns it as int   C:\Users\ken\AppData\Local\Temp\SNAGHTML3df723af.PNG |  |

* Write a main class, ***ComputeMethodsApp.java***, that performs the following:
  + The main method reads three inputs from the user as below, creates an instance of the **ComputeMethods** class and invokes the methods defined to display the results.
    - Fahrenheit temperature
    - Width of the triangle
    - Height of the triangle
  + A sample out is as below

Enter Fahrenheit: **100**

Enter Width of the triangle: **3**

Enter Height of the triangle: **2**

Temp 100(Fahrenheit) in Celsius is 37.7778 Hypotenuse of the triangle is 3.605551275463989 The sum of two dice values is 9

* Task B.
  + Create a **Student** class as a super class which contains:
    - Student’s name as string
    - Student’s id as int
  + Create a subclass from the Student class named **InternationalStudent** which contains:
    - Student’s name (inherited)
    - Student’s id (inherited)
    - Country name as string
    - Visa period in years as int
  + Create a class named ***OnlineTool*** which contains:
    - The name of the tool as string
  + Create another subclass from the Student class named **OnlineStudent** which contains:
    - Student’s name (inherited)
    - Student’s id (inherited)
    - Name of online-tool as OnlineTool (defined above)
    - Study period in years as int
  + Include appropriate methods in each class
    - No detailed statements in the methods are required for this task
  + The *InternationStudent* class will have a default constructor and an overloaded constructor which receives the visa period
  + Create a main class that demonstrates polymorphism.
    - Declare Student objects and show how Student objects can be either *OnlineStudent*

or *InternationStudent*.

* + - Demonstrate how to make a *Student* object behave as an *InternationalStudent* and how to make the same *Student* object behave as an *OnlineStudent*.

**Activity 6**

* Task A.
  + You are asked to write a program that calculates the total and the average of int numbers.
  + First, create a class named ***TotalAverageCalculator*** that contains two methods:
* public int getTotal(int[] nos)
  + It calculates the total of the given numbers in the array and returns it as int.
* public double getAverage(int[] nos)
  + It calculates the average of the given numbers in the array and returns it as double.
  + Write a main class, ***TotalAverageApp.java***, that uses the two methods defined above.
    - A sample main class.

public static void main(String[] args) {

TotalAverageCalculator tac = new TotalAverageCalculator(); int[] numbers = {10, 5, 6, 7, 89};

int total = tac.getTotal(numbers);

double average = tac.getAverage(numbers);

System.out.println("Total: " + total); System.out.println("Average: " + average);

}

* + - A sample result.

Total: 117

Average: 23.4

* Task B.
  + You created a program in a previous activity that separates a number to its individual digits, named *NumberWithSpaces*.
  + Modify the program so that it reads the number from a text file (named ***number.txt***) and write the separated digits in another text file (named ***separated.txt***).
    - The number from the text file is read as String, so the String number needs to be converted to int number

- E.g. "12345" to 12345

- Use ***Integer.parseInt()*** method. Refer to Java API document

* + - An example of *number.txt* file that your program reads.

12345

* + - An example of *separated.txt* file that your program writes.

1 2 3 4 5

* + Save the main program as ***NumberWithSpacesWithFiles.java***
* Task C.
  + You developed the *ComputeMethodsApp* program in the previous activity. Now, you need to test the program.
  + Identify testing requirements and prepare test plan document.
    - Review and record the test results in a test document.
    - Use the test report template provided.
  + Prepare unit testing environment with a unit testing framework, e.g. JUnit.
    - Perform unit testing the ***fToC()*** method.

**End of Assessment Tool**